Int. J. on Recent Trends in Engineering and Technology, Vol. 9, No. 1, July 2013

A Cryptographic Hardware Revolution in Communication Systems using Verilog HDL

[1]Jasyanth Koppula and [2]Bala Sindhuri Kandula
[1]M.Tech student, Department of Electronics and Communication Engineering, SRKR Engineering College, Bhimavaram, Andhra Pradesh, India
jasyanth.koppula@ yahoo.com
[2]Asst Professor, SRKR Engineering College, Department of Electronics and Communication Engineering, Bhimavaram, Andhra Pradesh, India
k.b.sindhuri@gmail.com

***Abstract*— Advanced Encryption Standard (AES), is an advancement of Federal Information Processing Standard (FIPS) which is an initiated Process Standard of NIST. The AES specifies the Rijndael algorithm, in which a symmetric block cipher that processes fixed 128 bit data blocks using cipher keys with different lengths of 128, 192 and 256 bits. The earliest Rijndael algorithm had the advantage of combining both data block sizes of 128, 192 and 256 bits with any key lengths. AES can be programmed in pure hardware Verilog HDL, Which includes Multiplexer to enhance more secure to Cipher text. The results indicate that the hardware implementation proposed in this project is Decrementing Utilization of resource and power consumption of 113 mW than other implementation. Using FPGA lead to reliability on source modulations. This project presents the AES algorithm with regard to FPGA and Verilog HDL. The software used for Simulation is ModelSim-Altera 6.3g_p1 (Quartus II 8.1). Synthesis and implementation of the code is carried out on Xilinx ISE 13.4 (XC6VCX240T) device is used for hardware evaluation.**

***Index Terms*- FIPS, FPGA, Modelsim software, NIST, Rijndael algorithm, Verilog HDL, Xilinx ISE.**

I. Introduction

In the recent years, Using FPGA in production versions of electronic systems increased. Advanced Encryption Standard (AES) FPGA [5] designs typically unroll the loops within the AES design followed by deep pipelining of a 128 Bit data path to achieve throughputs of the order of tens of gigabits per sec based on The National Institute of Standards and Technology (NIST) Standards. These designs have utility in applications such as hardware accelerator cards for e-Commercial servers & secure trunk communication.

This design for AES on FPGAs design [1]-[2], which is an 8 bit Application Specific Instruction Processor (ASIP) which supports key expansion (can be programmed for 128 bits) encipher and decipher. This design less than 40% of the resources of the smallest available Xilinx Virtex VI. This can be used in applications, which had low power and low Area as priorities design [2].

An ASIP capable of performing AES encipher and decipher operations using a truly 8-bit data path. The design utilizes a novel version of the *Sub- Bytes* operation using existing composite field arithmetic[9], However, the three required multiplications are performed using a single resource shared multiplier with the commensurate area saving.

The ASIP achieves an average encipher-decipher throughput of 3.1Mbps and utilizes less than the one third of the resources of smallest Xilinx Virtex VI (XC6VCX240T).

The results presented are for Xilinx FPGAs, however, the optimizations made are equally applicable to other vendor's FPGAs. The comparison between FPGA designs [3] which incorporate ROMs and those which do not is sometimes problematic. Here, this is solved by converting the amount of block memory used into an equivalent number of slices. This yields a single area figure for any design. The throughput–area ratio is frequently used as the academic measurement of design efficiency, however, there are economic and engineering savings to be made by striving towards the lowest possible area design which meets the overall system requirements. The designs in this paper are aimed at challenging the lowest area end of the design space while still providing a usable throughput.

II. Description of AES Algorithm

The algorithm is composition of three main parts Cipher Text , Inverse Cipher Text, and Key Expansion. The Cipher converts normal text to an unintelligible form called cipher text while Inverse Cipher converts data back into its original normal text form called plaintext. The Key Expansion generates a Key Schedule that is used in Cipher and Inverse Cipher procedure. The Cipher and Inverse Cipher are composed of specific number of rounds shown below (Table 1). For the AES algorithm design [4], the number of rounds to be performed during the execution of the algorithm is dependent on the key length.

The sequence in which the operation is carried out is as follows:

Round 1**:**

A. Add Round key.

Following Rounds**:**

A. Sub Bytes.
B. Shift Rows.
C. Mix Column.
D. Multiplexer.
E. Add Round Key.

DOI: 01.IJRTET.9.1.11

TABLE I: NUMBER OF ROUNDS IN AES ALGORITHM

Algorithm	Block Size (Nb Words)	Key Length (Nk Words)	Number of Rounds (Nr)
AES-128-bits key	4	4	10
AES-192-bits key	4	6	12
AES-256-bits key	4	8	14

Final Round**:**

A. Sub Bytes.

B. Shift Row.

C. Add Round Key

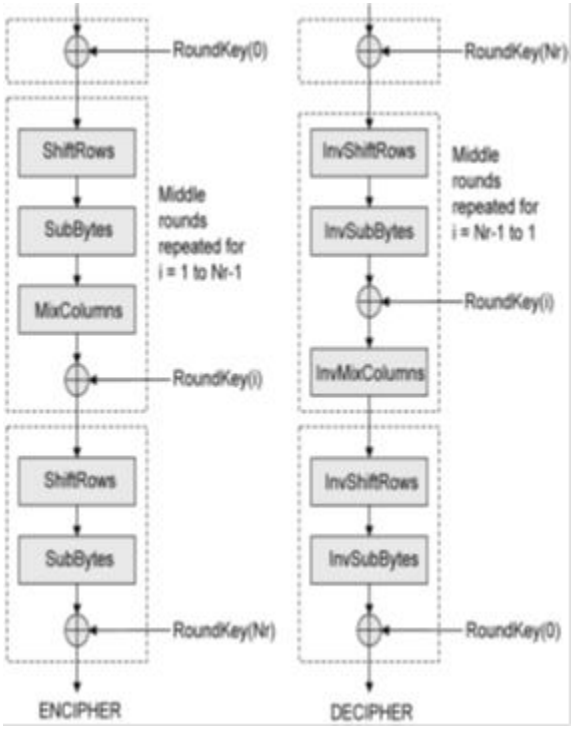

Fig 1: Basic Concept of the Algorithm

This is shown in figure 1. The AES algorithm can be implemented in both hardware and software. The software implementation of AES algorithm is a slow process when compared with hardware process.

III. AES ENCRYPTION PROCESS

Briefly, this block cipher can perform encipher and decipher operations using the repeated operation of a Substitute Permute Network (SPN) on 128 bits of data.

Each time the SPN is used it is supplied with a different ***RoundKey*****.** These are generated by a function known as ***KeyExpansion*****.** Three different key lengths were specified, 128, 192 and 256 bits. Which in turn require 10, 12 and 14 rounds of substitution and permutation. The first and final rounds differ from the middle rounds and the overall process is summarized in **Fig. 1**. The AES specification provides two alternative designs [8] for decipherment. In this design, the one where the ***Round-Keys*** are the same as encipher was selected. These are applied in reverse order for decipherment.

A. Sub Bytes:

In the sub bytes stage the data in the plain text form is substituted by some predefined values from a substitution box which is an invertible form [9].

B. Shift Rows:

In shift rows operation the rows in the 4×4 matrix is shifted to left r bits and r varies with the rows of the matrix(r=0 for row1, r =1 for row2, r =2 for row3, r =3 for row 4). This process is illustrated in figure 2.

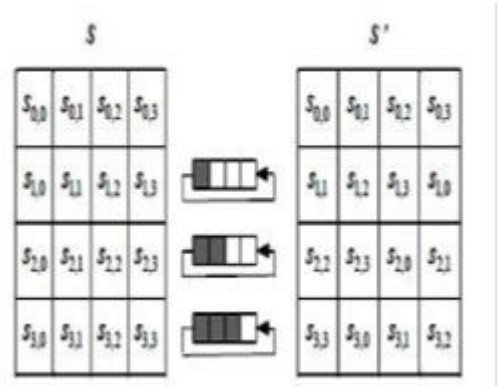

Fig 2: Shift Rows

C. Mix Columns

MixColumns is calculated using the below formula. Here a_0, a_1, a_2, a_3 is calculated using the polynomials as below

$$a(x) = \{2\}x^3 + \{3\}x^2 + \{1\}x + 1 \qquad (1)$$

The MixColumns transformation operates on the step column by column, generating each column as a four term polynomial as in **Figure 3**. The Columns are assumed as polynomials over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial a(x) which is got from the above formula. This can also written as a matrix multiplication

$$s'(x) = a(x) \otimes c(x) \qquad (2)$$

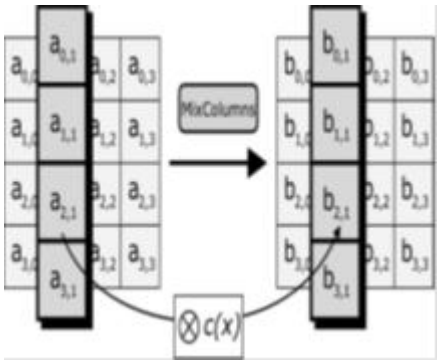

Fig 3: Mix Column

DOI: 01.IJRTET.9.1.11

D. Multiplexer 2:1:

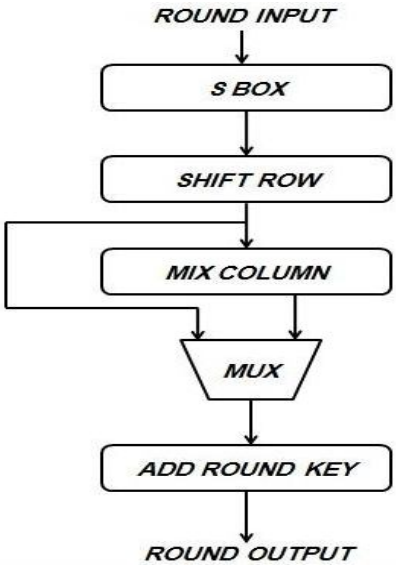

Fig 4: Single round of AES algorithm using MUX

In this Algorithm Multiplexer is used between Shift Row and Mix Columns as in **Figure 4**. This is design [10] used to enhanced the security of the cipher text and also to ensure the quick processing of the cipher text.

E. Add Round Key:

In the add round key step the 128 bit data is XORed with the sub key of the current round using the key expansion operation. The add round key is used in two different places one during the start that is when round r = 0 and then during the other rounds that is when 1 d• round d• Nr, where Nr is the maximum number of rounds. The formula to perform the add round key is

$$S'(x) = S(x) \otimes R(x)$$

S'(x) – state after adding round key

S(x) – state before adding round key

R(x) – round key

F. Key Expansion:

The key expansion has three steps:

a) Byte Substitution *subword* ( )

b) Rotation *rotword* ( )

c) XOR with RCON (round constant)

The input to key schedule is the cipher key K. Key expansion generates a total of $N_b (N_r + 1)$ words as shown in **Figure 5**. The algorithm requires an initial set of N_b words and each of the N_r rounds requires N_b words of key data. The obtained key schedule consists of a linear array of 4-byte words, denoted $[w_i]$, with i in the range 0 d" $i < N_b (N_r + 1)$. The subword( ) function takes a four byte input and applies the byte substitution operation and produces an output word. The rotword ( ) takes a word [a0, a1, a2, a3] as input and performs a cyclic permutation to produce [a1, a2, a3, a0] as output word. The round constant word array rcon [i] is calculated using the below formula in finite field.

$$rcon[i] = x^{(254+i)} \bmod x^8 + x^4 + x^3 + x + 1 \quad (3)$$

The first N_k words of the expanded key are filled with the cipher key. Every word w[i] is equal to the XOR of previous word w[i-1] and the word N_k positions earlier w[i-N_k]. For the words in positions that are a multiple of N_k, a transformation is applied to w[i-1] prior to the XOR and followed by an XOR with a round constant Rcon[i]. This transformation contains a cyclic shift of the bytes in a word rotword() and byte substitution subword(). But in key expansion of 256-bit cipher if Nk=8 and i-4 is a multiple of Nk then subword() function is applied to w[i-1] prior to the XOR.

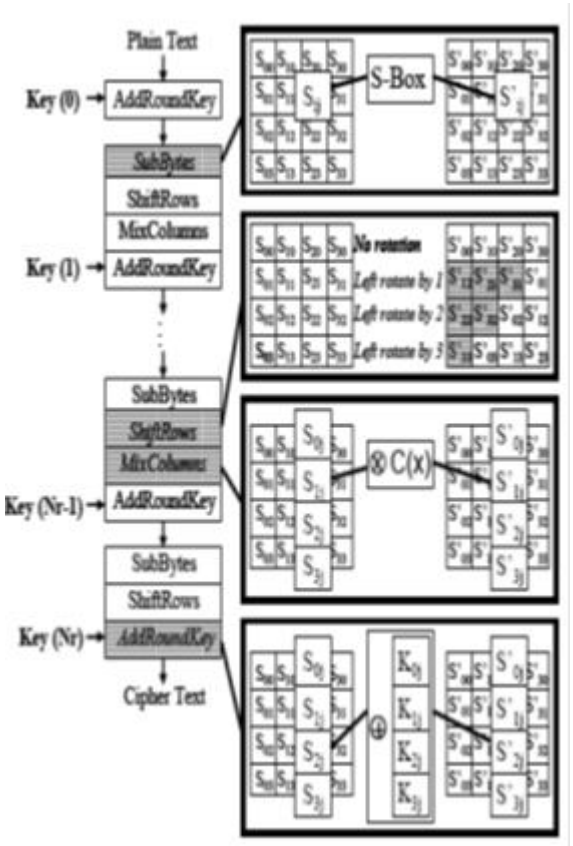

Fig 5: AES Encryption Process

IV. AES Decryption Process

The decryption of the data which was encrypted using the AES is done by inverting all the encryption operations with the same key with which it is encrypted since the AES is a symmetric encryption standard. In the design [4], decryption process the sequence of the transformations differs from that of the encryption but the key expansion for encryption and decryption are the same. The several properties of the AES algorithm [7] allow for an equivalent decryption with the same sequence of transformations as that in encryption.

The operations of the decryption process are listed below

A. Inverse Sub Bytes.

B. Inverse Shift Rows.

C. Add Round Key.

D. Inverse mix columns.

DOI: 01.IJRTET.9.1.11

*A. Inverse Sub Bytes***:**

This operation is similarly as it is in the encryption process but the only difference is the inverse of the substitution box is used here since the substitution box which we used in the encryption is invertible.

B. Inverse Shift Rows:

The inverse shift rows operation is an inverse process of the shift row operation in the encryption process by right shifting the elements in the rows.

C. Add Round Key:

The add round key process is as same as that of the one in the encryption process.

D. Inverse Mix Columns:

In inverse mix column operation is the same operation in the mix column is done but with the different matrix as in **Figure 6**.

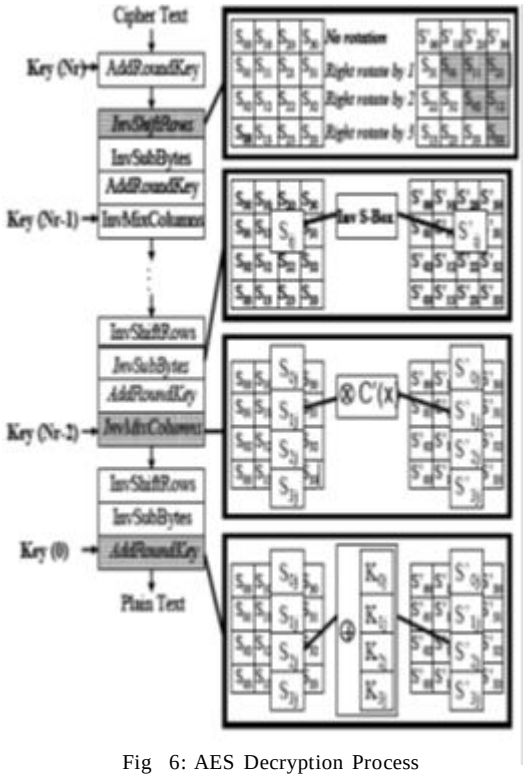

Fig 6: AES Decryption Process

V. Implementation

The AES algorithm is implemented using Verilog HDL coding in Xilinx ISE 8.1. First, the algorithm is Simulated using ModelSim by encrypting and circuit Diagram is obtained as shown in **Figure 7** and decrypting a single 128 bit block and Synthesizing and implementation of the code is carried out on **Xilinx ISE 13.4** device. Then the key is expanded to use for 192, 256 bit blocks. The Power Consumption is 63% and Utilization is more than the previous projects [2], [6], which is double than other implementations. The implementation output is shown in **Table II**

Fig 7: Circuit diagram for AES algorithm

Table II : AES-128 Implementation Results

DESIGN	DEVICE	FREQUENCY(MHZ)	AREA	THROUGHPUT(MBPS)
AES-128	VIRTEX 5	174.166	2,499	2229
AES-128	VIRTEX 6	200.087	2,715	4998
AES-128	VIRTEX 4	75.850	6,076	4677
AES-128	SPARTAN 6	249.236	5,067	4012

VI. Testing and Verification

To ensure the proposed design gives better results in terms of Utilization and Power Consumption the design is implemented Xilinx Virtex VI **(XC6VCX240T) and** FPGA device used for downloading. The device Comparison table of algorithm as shown in **Table III** i.e. AES-128 in same hardware is shown.

Table III: Comparison with other designs

Design	*Throughput (MBPS)*	*Area (CLBs)*	*Throughput/ area*
AES 128	*4998*	*2715*	*1.840*
[1],[6]	*1163*	*6701*	*0.173*
[2]	*353*	*3328*	*0.106*

The power consumption of the device for the algorithm i.e. AES-128 is 113 mW on the same hardware.

Input is taken as text data which is also known as plaintext. Here the plaintext is encrypted with the help of key. Finally the encrypted data obtained in unknown form as shown in **Figure 8**.

Key: 000102030405060708090a0b0c0d0e0f
Input: 0a940bb5416ef045f1c39458c653ea5a
Output: XZXZ2a502ed505bbc117c70e5163194f

The Encrypted data is given as the input to the decryption block which will gives the original Plaintext as the output. The output obtained by using Modelsim 6.1 is shown in **Figure 9**.

Key: 000102030405060708090a0b0c0d0e0f
Input: XZXZ2a502ed505bbc117c70e5163194f
Output: 0a940bb5416ef045f1c39458c653ea5a

Throughput Area is 4998 MBPS
Power Consumption is 113mW

DOI: 01.IJRTET.9.1.11

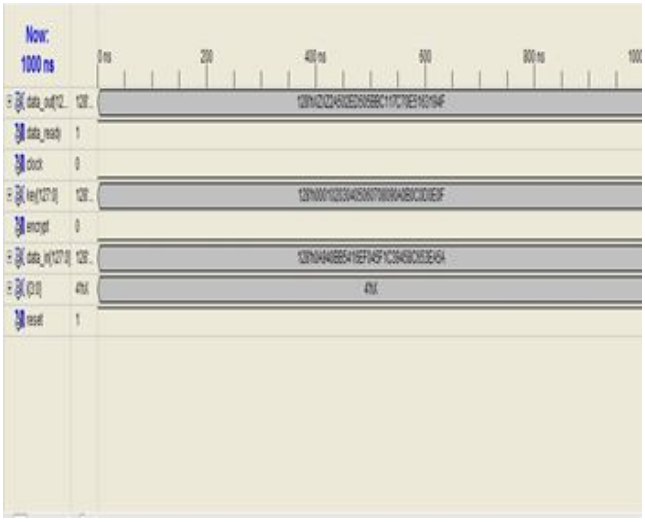

Fig 8: Wave forms of Encryption using Xilinx

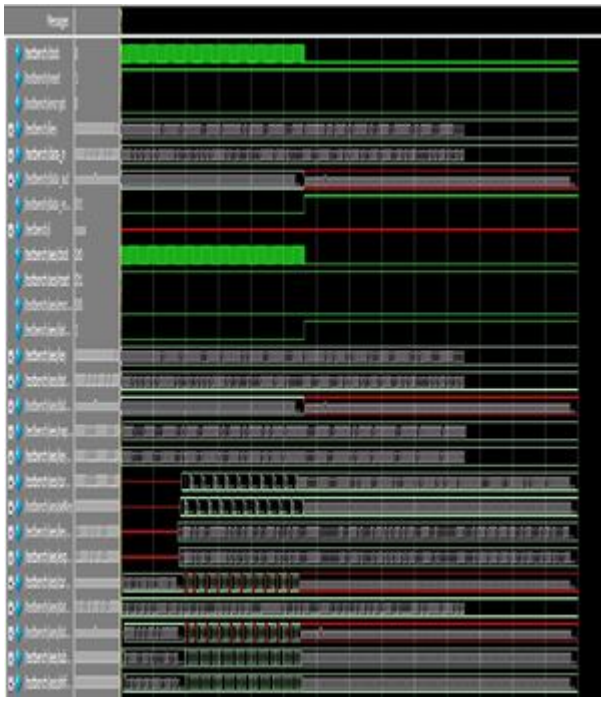

Fig 9: Output using Modelsim 6.1

VII. Conclusion

The Advanced Encryption Standard algorithm is an iterative private key symmetric block cipher that can process data blocks of 128 bits through the use of cipher keys with lengths of 128, 192, and 256 bits. Here Focus is on implementing AES algorithm using the reconfigurable hardware technology based on Field Programmable Gate Arrays (FPGA) in Verilog HDL. The architecture of an iterative AES and Deep pipeline architecture of algorithm is presented.

The algorithm can accept data and keys of 128 bits. It can achieve a maximum throughput of 4998 MBPS. The achieved Power consumption is about four times lesser than other methods [1], [2], [6]. reported A less power Consumption of 113mW is achieved using this architecture. Optimized and Synthesizable Verilog code is developed for the implementation of both encryption and decryption process using Xilinx Family. Each program is tested with some of the sample vectors provided by NIST. The combination of security, and low power consumption implementation makes it a very good choice for wireless communication systems.

Appendix A: Terms And Definitions

The following definitions are used throughout the algorithm:

AES: Advanced Encryption Standard Affine a transformation consisting of multiplication by a matrix followed Transformation by the addition of a vector

Array: An enumerated collection of identical entities (e.g., an array of bytes).

Bit: A binary digit having a value of 0 or 1.

Block: Sequence of binary bits that comprise the input, output, State, and Round Key. The length of a sequence is the number of bits it contains. Blocks are also interpreted as arrays of bytes.

Byte: A group of eight bits that is treated either as a single entity or as an array of 8 individual bits.

Cipher: Series of transformations that converts plaintext to ciphertext using the Cipher Key. Cipher **Key**: Secret, cryptographic key that is used by the Key Expansion routine to generate a set of Round Keys; can be pictured as a rectangular array of bytes, having four rows and ***Nk*** columns.

Ciphertext: Data output from the Cipher or input to the Inverse Cipher.

Inverse Cipher: Series of transformations that converts ciphertext to plaintext using the Cipher Key.

Key Expansion: Routine used to generate a series of Round Keys from the Cipher Key.

Plaintext: Data input to the Cipher or output from the Inverse Cipher.

Rijndael: Cryptographic algorithm specified in this Advanced Encryption Standard (AES).

Round Key: Round keys are values derived from the Cipher Key using the Key Expansion routine; they are applied to the State in the Cipher and inverse Cipher.

State: Intermediate Cipher result that can be pictured as a rectangular array of bytes, having four rows and ***Nb*** columns.

S-box: Non-linear substitution table used in several byte substitution transformations and in the Key Expansion routine to perform a one for-one substitution of a byte value.

Word: A group of 32 bits that is treated either as a single entity or as an Array of 4 bytes.

APPENDIX B: Algorithm Parameters, Symbols, and Functions

The following algorithm parameters, symbols, and functions are used throughout this standard:

Add RoundKey () : Transformation in the Cipher and Inverse Cipher in which a Round Key is added to the State using an XOR operation. The length of a Round

DOI: 01.IJRTET.9.1.11

Key equals the size of the State (i.e. for *Nb* =4, the Round Key length equals 128 bits/16 bytes).

InvMix Columns (): Transformation in the Inverse Cipher that is the inverse of MixColumns().

InvShift Rows (): Transformation in the Inverse Cipher that is the inverse of ShiftRows().

InvSub Bytes (): Transformation in the Inverse Cipher that is the inverse of SubBytes().

***K*:** Cipher Key.

Mix Columns (): Transformation in the Cipher that takes all of the columns of the State and mixes their data (independently of one another) to Produce new columns.

N_b: Number of columns (32-bit words) comprising the State. For this standard, ***Nb*** = 4.

N_k: Number of 32-bit words comprising the Cipher Key. For this standard, ***Nk*** = 4, 6, or 8.

N_r: Number of rounds, which is a function of ***Nk*** and N_b(which is fixed). For this standard, ***Nr*** = 10, 12, or 14.

Rcon[]: The round constant word array.

RotWord (): Function used in the Key Expansion routine that takes a four-byte word and performs a cyclic permutation.

ShiftRows (): Transformation in the Cipher that processes the State by cyclically shifting the last three rows of the State by different offsets.

SubBytes (): Transformation in the Cipher that processes the State using a Nonlinear byte substitution table (S-box) that operates on each of the State bytes independently.

SubWord (): Function used in the Key Expansion routine that takes a four-byte input word and applies an S-box to each of the four bytes to produce an output word. XOR Exclusive-OR operation.

REFERENCES

[1] Mohammed Benaissa, "Very Small FPGA Application-Specific Instruction Processor for AES", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 53, NO. 7, JULY 2006,pp.1477-1486.

[2] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 9, pp. 957–967, Sep. 2004.

[3] A. Hodjat and I. Verbauwhede, "A 21.54 Gbits/s Fully Pipelined AES Processor on FPGA," in Proc. FCCM'04, Apr. 2004, pp. 308–309.

[4] J. Zambreno, D. Nguyen, and A. Choudhary, "Exploring Area/ Delay Trade-offs in an AES FPGA Implementation," in Proc. LNCS FPL'04, Antwerp, Belgium, 2004, vol. 3203, pp. 575–585.

[5] N. Pramstaller and J. Wolkerstorfer, "A universal and efficient AES co-processor for field programmable logic arrays," in Proc. LNCS FPL'04, 2004, vol. 3203, pp. 565–574.

[6] Dandalis A., Prasanna V.K, Rolim J.D, "A Comparative Study of Performance of AES Final Candidates Using FPGAs", Cryptographic Hardware and Embedded Systems Workshop (CHES 2000), Worcester, Massachusetts, 2000

[7] G. Rouvroy, F. X. Standaert, J. Quisquater and J. D. Legat, "Compact and efficient encryption/ decryption module for FPGA Implementation of the AES Rijndael very well suited for small embedded applications," in Proc. ITCC'04, Apr. 2004, vol. 2, pp. 583–587.

[8] V. Fischer and M. Drutarovsky, "Two Methods of Rijndael Implementation in Reconfigurable Hardware," in Proc. CHES'01, 2001, vol. 2162, pp. 77–92.

[9] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture With S-Box Optimization," in Proc. LNCS SIACRYPT'01, Dec. 2001, vol. 2248, pp. 239–254.

[10] F. X. Standaert, G. Rouvroy, J. Quisquater, and J. Legat, "A Methodology to Implement Block Ciphers in Reconfigurable Hardware and its Application to Fast and Compact AES RIJNDAEL," in Proc. ACM FPGA'03, Monterey, CA, 2003, pp. 216–224.

DOI: 01.IJRTET.9.1.11